VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvovanie individuálnej odbornej praxe

Individual Professional Practice in the Company

2015

Mário Červeň

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Mário Červeň**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Railsformers s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Pavla Dráždilová, Ph.D.**

Konzultant bakalářské práce: Ing. Lukáš Kamp

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlásenie študenta

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne. Uviedol som všetky literárne zdroje a publikácie z ktorých som čerpal.

V Ostrave 7. mája 2015

.....

podpis študenta

Prehlásenie zástupcu spolupracujúcej právnickej alebo fyzickej osoby

Súhlasím so zverejnením tejto bakalárskej práce podľa požiadaviek čl. 26, odst. 9 Študijného a skúškového rádu pre štúdium v bakalárskych programoch VŠB-TU Ostrava.

Railsformers s.r.o.
IČ: 24704440 DIČ: CZ24704440
www.railsformers.com
info@railsformers.com

V Ostrave 7. mája 2015

.....

podpis zástupcu

Touto cestou chcem poďakovať Mgr. Pavle Dráždilovej, Ph.D. za rady a ústretový prístup pri písaní tejto práce, ďalej môjmu konzultantovi Ing. Lukášovi Kampovi za pomoc pri absolvovaní praxe a Ing. Jiřímu Kubicovi za umožnenie praxe vo firme Railsformers s.r.o.

Abstrakt

Predmetom tejto bakalárskej práce je prax, ktorú som vykonával vo firme Railsformers s.r.o. sídliacej v Ostrave. Firma sa zameriava na tvorbu webových aplikácií vo webovom frameworku Ruby on Rails. Konkrétne sa firma zameriava na podnikové riešenia, sociálne siete, komunitné portály a iné komplexné systémy. Cieľom tejto práce je popis práce, ktorú som vykonával v tejto firme a zoznámenie sa s možnosťami frameworku Ruby on Rails pre vývoj webových aplikacií. Zameral som sa na tvorbu informačných systémov s využitím najpoužívanejších technológií.

Kľúčové slová: Ruby, Ruby on Rails, prax, webová aplikácia, informačný systém, HTML, CSS, JavaScript

Abstract

My bachelor's thesis is concerning my work experience at Railsformers s.r.o. company residing in Ostrava. The company's aim is to make web applications in Ruby on Rails framework. The company specializes in enterprise solutions, social networks, community portals, and other complex systems. The aim of this thesis is the work I did at this company and a familiarization with Ruby on Rails framework for making web applications. I focused on creating information systems with the help of the most popular technologies.

Keywords: Ruby, Ruby on Rails, practice, web application, information system, HTML, CSS, JavaScript

Zoznam použitých skratiek a symbolov

CSS	–	Cascading Style Sheets
HTML	–	Hyper Text Markup Language
MVC	–	Model View Controller
PDF	–	Portable Document Format
UI	–	User Interface – užívateľské rozhranie
WWW	–	World Wide Web

Obsah

1 Úvod **2**

2 Firma **3**

3 Použité technológie **4**
3.1 Ruby . . . 4
3.2 Návrhové a architektonické vzory . . . 4
3.3 HTML, CSS, JavaScript . . . 5
3.4 Front-end, Back-end . . . 7
3.5 Bootstrap . . . 8
3.6 Font Awesome . . . 8
3.7 Mailcatcher . . . 9
3.8 Git . . . 9

4 Pracovná náplň praxe **10**
4.1 Harmonogram praxe . . . 10
4.2 Oboznámenie sa s prostredím . . . 10
4.3 Inštalácia Ruby, Ruby on Rails . . . 10
4.4 Cvičná úloha . . . 11
4.5 Projekt Hedurio . . . 11
4.6 Projekt Topstone . . . 12
4.7 Projekt požičovňa krojov Ostravica . . . 12
4.8 Projekt MPM . . . 18

5 Záver **24**

6 Referencie **25**

7 Zoznam príloh **27**

1 Úvod

K bakalárskej práci som mal na výber medzi klasickou prácou a praxou vo firme. Vybral som si prax vo firme, pretože to je spôsob ako získať potrebné zručnosti a znalosti, ktoré mi v budúcnosti pomôžu pri uplatnení sa v práci.

Odbornú prax som vykonával vo firme Railsformers s.r.o. sídliacej v Ostrave. Firmu som si vybral na základe jej dobrých referencií. Taktiež mi ponúkla možnosť naučiť sa pre mňa dovtedy neznámy programovací jazyk.

Vo firme som pracoval na pozícii Ruby on Rails programátor. Vykonával som zadané programovacie úlohy na tvorbu informačných systémov a webových aplikácií. Samostatne som vytvoril informačný systém pre správu a vypožičiavanie krojov (projekt požičovňa krojov Ostravica) a informačný systém pre správu aktivít zamestnancov spoločnosti MgC group (projekt MPM). Ďalej som vykonával aj menšie úlohy ako úprava vzhľadu a preklad už hotových webových aplikácií. Podieľal som sa na tvorbe informačného systému pre riadenie bezpečnostných agentúr (projekt Hedurio) a informačného systém na predaj a prezentáciu značkových kamenných povrchov (projekt Topstone).

U úloh zameraných na tvorbu informačných systémov bolo mojou úlohou analyzovať požiadavky zákazníka, navrhnúť systém podľa daných požiadaviek a následne ho naimplementovať. U úloh na preklad aplikácií som vytváral lokalizáciu do jazyka, v ktorom aplikácia dovtedy nebola preložená.

Táto práca je rozdelená do nasledovných kapitol:

- **Firma** – popisuje firmu, v ktorej som pracoval.
- **Použité technológie** – popisuje jednotlivé technológie, ktoré som pri práci používal.
- **Pracovná náplň praxe** – obsahuje harmonogram praxe a popis jednotlivých úloh, ktoré som vykonával.
- **Záver** – obsahuje zhodnotenie mojej bakalárskej praxe a popis získaných znalostí a skúseností.

2 Firma

Firma Railsformers s.r.o. [1] je firma sídliaca v Ostrava zameraná na vývoj internetových aplikácií a informačných systémov. Pre tvorbu projektov využívajú framework Ruby on Rails, s ktorým majú dlhoročné skúsenosti. Špecializujú sa na projekty väčšieho rozsahu, ako sú napr. enterprise riešenia, sociálne siete, komunitné portály a iné komplexné systémy. Cieľom firmy je uspokojiť požiadavky zákazníka a naviazať s ním dlhodobý pracovný vzťah.

Pri vývoji používajú okrem Ruby on Rails aj AJAX a rôzne knižnice ako napríklad jQuery UI. K optimalizácii pre vyhľadávače používajú SEO (z angl. search engine optimization).

Posledné práce firmy:

- Hedurio - komplexný online informačný systém pre riadenie bezpečnostných agentúr.
- sMoneybox - online účtovníctvo pre správu osobných financií. Aktuálne má 15 800 registrovaných užívateľov.
- sÚčto - online aplikácia pre fakturáciu a účtovníctvo
- sRecepty - kulinárské stránky s receptmi. Umožňuje zdieľanie receptov medzi užívateľmi. Aktuálne má viac ako 3000 aktívnych užívateľov.

3 Použité technológie

V tejto časti budú uvedené technológie, ktoré som počas praxe používal.

3.1 Ruby

Ruby [2] je interpretovaný skriptovací programovací jazyk zameraný na jednoduchosť a produktivitu. Vďaka svojej jednoduchej syntaxy je aj jednoduchý na učenie. Ďalšou z jeho vlastností je, že je plne objektovo orientovaný – všetko v Ruby je objekt. Výhodnou vlastnosťou je aj jeho multiplatformovosť.

Jazyk Ruby vytvoril Japonec Yukihiro Matsumoto. Prvá verzia bola uverejnená v roku 1995.

```
cities = %w[ London Oslo Paris Amsterdam Berlin ]
visited = %w[ Berlin Oslo ]

puts "Rozdiel poli cities - visited: ", cities - visited
```

Výpis 1: Ukážka práce s polom v Ruby – môžme získať rozdiel dvoch polí

3.1.1 Ruby on Rails

Ruby on Rails [3] je open-source webový framework založený na programovacom jazyku Ruby. Optimalizovaný pre jednoduchý vývoj. Zvýhodňuje konvencie nad konfiguráciou čím zabezpečuje výbornú čitateľnosť kódu. Vytvoril ho dánsky programátor David Heinemeier Hansson pri práci na projekte Basecamp. Vydaný bol 13. decembera 2005. Využíva softvérový architektonický vzor Model View Controller 3.2.1.

3.1.2 RubyGems

RubyGem (skrátene gem) je knižnica používaná v Ruby on Rails. Zabezpečuje rozšíriteľnosť Ruby on Rails. Zoznam využívaných gemov v Ruby on Rails aplikácii sa nachádza v súbore nazývanom Gemfile.

3.2 Návrhové a architektonické vzory

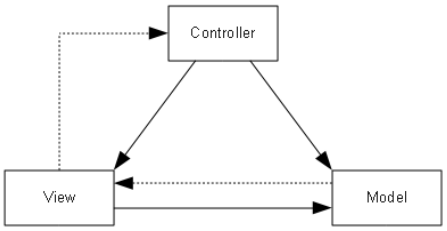
Ruby on Rails využíva vzory Model View Controller a Active Record.

3.2.1 Model View Controller

Model View Controller (MVC) [4, str. 286] je softvérový architektonický vzor. Často je používaný vo webových frameworkoch, vrátane Ruby on Rails.

MVC vyžaduje rozdelenie do 3 komponent:

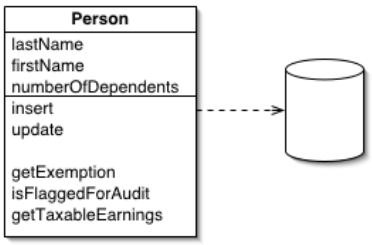
- **Model** je objekt, ktorý reprezentuje informácie o doméne (oblasť činnosti, ktorú riešime).
- **View** reprezentuje zobrazenie dát modelu v užívateľskom rozhraní.
- **Controller** sa stará o zmeny v informáciách. Získava užívateľský vstup, manipuluje s modelom a aktualizuje view.



Obrázok 1: Vzťahy medzi modelom, view a controllerom [5]

3.2.2 Active Record

Vo svojej podstate je to „objekt, ktorý obaľuje riadok v databázovej tabuľke či view, zapúzdruje prístup k databáze a pridáva k týmto dátam doménovú logiku“ [4, str. 147]. Objekt sa teda stará o dáta aj logiku čo pri veľkej komplexnosti môže spôsobovať problémy s prehľadnosťou. Dáta sú typicky uložené v databáze.



Obrázok 2: Active Record [4, str. 147]

3.3 HTML, CSS, JavaScript

HTML, CSS, JavaScript tvoria dnešné webové stránky. Pri práci som pre zjednodušenie HTML syntaxe používal väčšinou HAML.

3.3.1 HAML

HAML [6] zjednodušuje syntax v HTML (prípona .html) a ERB (Embedded Ruby, prípona .erb). V HAML sa nepíšu koncové značky a samotné značky majú skrátený zápis. Záleží však na odsadení. Konečný kód je tak kratší, prehľadnejší a ľahší na čítanie. U súborov sa používa prípona .haml.

HTML/ERB syntax	**HAML syntax**
nadpis	%h1 nadpis
	.content
<% if condition %>	- if condition
<%= user.name %>	= user.name
Hello World!	%strong{id: "message"} Hello World!

Tabuľka 1: HTML/ERB syntax vs HAML syntax

```
<section class="container">
  <h1><%= post.title %></h1>
  <h2><%= post.subtitle %></h2>
  <div class="content">
    <%= post.content %>
  </div>
</section>
```

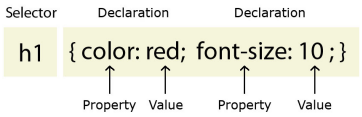
Výpis 2: Príklad v HTML/ERB

```
%section.container
  %h1= post.title
  %h2= post.subtitle
  .content
    = post.content
```

Výpis 3: Ten istý príklad v HAML

3.3.2 CSS

CSS (Cascading Style Sheets) je spolu s JavaScriptom neoddeliteľná súčasť dnešných webov. CSS udáva vzhľad HTML elementov. Jeho cieľom je oddelenie vzhľadu dokumentu od jeho štruktúry a obsahu. Verzia CSS3 je súčasťou HTML5.



Obrázok 3: Pravidlá písania CSS [7]

3.3.3 JavaScript

JavaScript [8] je skriptovací programovací jazyk (interpretovaný, spracováva sa priamo zdrojový kód) určený pre riešenie dynamiky WWW stránok na strane klienta. Sú ním obvykle ovládané prvky GUI, animácie, efekty a podobne. Používa sa vo všetkých moderných weboch ako súčasť zdrojového kódu HTML.

Vlastnosti JavaScriptu:

- je multiplatformový
- je závislý na interpretačnom prostredí (prehliadači)
- je objektovo orientovaný, ale beztriedny
- je case-senzitívny
- syntaxou je podobný jazykom typu C/C++/Java (niekedy voľnejší, napr. nepoužíva „;“ za príkazom)
- je beztypový

Používal som aj knižnicu JavaScriptu – **jQuery** pre ľahší vývoj a implementáciu.

```
b1 = document.getElemementById("
    button1")
b1.addEventListener('click', function(){
    alert("Hello");
});
```

Výpis 4: Príklad kódu napísanom v JavaScripte. Po kliknutí na tlačidlo s id="button1" vyskočí upozorňovacie okienko s textom.

```
$('#button1').click(function(){
    alert("Hello");
});
```

Výpis 5: Ten istý príklad napísaný v jQuery

Zároveň som používal aj **CoffeeScript** [9]. Je to programovací jazyk, ktorý sa prekladá do JavaScriptu. Má syntax inšpirovanú jazykmi Ruby, Python, Haskell pre stručnosť a jednoduchšiu čitateľnosť kódu, preto sa v Ruby on Rails aplikáciách často využíva. Namiesto kľúčového slova function používanom v JavaScripte používa symbol „->“.

```
function gcd(x, y) {
    var z
    do {
        z = x % y
        x = y
        y = z
    } while (y != 0)
    return x
}
```

Výpis 6: Príklad na funkciu v JavaScripte pre výpočet najväčšieho spoločného deliteľa

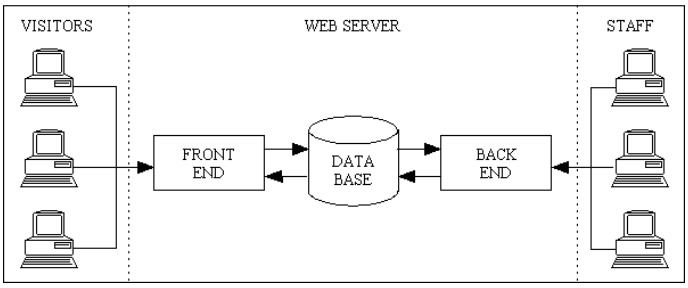
```
gcd = (x, y) ->
    [x, y] = [y, x%y] until y is 0
    x
```

Výpis 7: Ten istý príklad v CoffeeScripte

3.4 Front-end, Back-end

Ako **front-end** [10] (alebo tiež front end, frontend) sa označuje prostredie webu viditeľného bežným návštevníkom.

Back-end (alebo tiež back end, backend) je prostredie webovej aplikácie slúžiace k administrácii. Z pravidla sa tu nachádzajú rôzne administrátorské nastavenia (napr. manipulácia so štruktúrou celého webu, pridávanie ďalších administrátorov) a obsah, ktorý front-end zobrazuje. Prístup k nemu majú len oprávnení používatelia danej webovej aplikácie. Pri back-ende sa často nekladie tak veľký dôraz na vzhľad a prístupnosť ako pri front-ende.



Obrázok 4: Front-end, Back-end [11]

3.5 Bootstrap

Bootstrap [12] je HTML, CSS, JavaScript framework, ktorý slúži na vytváranie responzivných web stránok a uľahčuje tvorbu dizajnu. K používaniu je zdarma a je teda ľahko dostupný a často používaný.

Bootstrap využíva mriežkový systém, ktorí uľahčuje umiestňovanie prvkov v UI. Webová stránka je tak vertikálne rozdelená na riadky, horizontálne na 12 stĺpcov.

```
.row
  .col-sm-6
    %h3 Column 1
  .col-sm-6
    %h3 Column 2
    %p Lorem ipsum dolor sit amet
```

Výpis 8: Ukážka Bootstrap mriežkového systému v HAML

3.6 Font Awesome

Font Awesome [13] je štýl písma (font), ktorý používa namiesto písmen rôzne ikony. Keďže je to štýl písma, tak si ho môžme jednoducho prispôsobovať pomocou CSS (napr. farba, veľkosť, tieň). Nevyžaduje JavaScript a je teda jednoducho použiteľný. Aktuálne využíva 519 ikon. Pomocou Font Awesome je možné aj kombinovať a prekrývať ikony.

Na otáčanie ikon slúžia triedy `fa-rotate-*` a `fa-flip-*` .

```
<i class="fa fa-shield fa-rotate-90"></i> otocenie o 90 stupnov
<i class="fa fa-shield fa-flip-horizontal"></i> horizontalne otocenie
```

Výpis 9: Otáčanie ikon

Ďalej sú tu aj animované ikony. Pre príklad kombináciou tried `fa-refresh` a `fa-spin` vznikne otáčacia sa ikona refresh, kombináciou tried `fa-spinner` a `fa-pulse` vznikne pulzujúca ikona spinner.

3.7 Mailcatcher

Mailcatcher [14] je jednoduchý server, ktorý zachytáva emaily odoslané z práve používaného počítača. Slúži tak na testovanie odosielaných mailov z tohto počítača. Emaily zobrazuje vo vlastnom užívateľskom rozhraní.

Aby sme mohli odosielané emaily testovať v Rails aplikácii, musíme nastaviť, aby sa emaily odosielané z počítača odosielali z portu 1025.

```
config.action_mailer.delivery_method = :smtp
config.action_mailer.smtp_settings = { address: "localhost", port: 1025 }
```

Výpis 10: Nastavenie odosielania mailov v Rails aplikácii

Príkazom `mailcatcher` zadaným do terminálu spustíme Mailcatcher server. Rozhranie Mailcatcheru nájdeme na http://localhost:1080/. V tomto rozhraní sa zobrazujú odoslané emaily.

3.8 Git

Git [15] je distribuovaný systém riadenia verzií. Je to vlastne systém na ukladanie zmien v projekte výhodný hlavne keď sa na projekte podieľajú viacerí používatelia. Zdieľajú tak jeden projekt, na ktorom môžu pracovať súčasne. Git vytvoril Linus Torvalds, pôvodne pre jadro Linuxu. Vydaný bol 7.4.2005.

Najdôležitejšou funkciou v Git je vetvenie. Pomocou vetvenia je možné vytvorenie viacerých vetiev, ktoré sú od seba nezávislé. Vetvy sa dajú vytvárať, zlučovať, odstraňovať. Používateľ si tak môže vytvoriť vetvu, v ktorej bude riešiť vývoj a zlúčiť ju do hlavnej vetvy až vtedy, keď bude daná vetva optimalizovaná.

4 Pracovná náplň praxe

V tejto časti bližšie popíšem postupy a činnosti, ktoré som vykonával pri práci na jednotlivých projektoch.

4.1 Harmonogram praxe

Časový priebeh mojej praxe:

- **1. deň:** oboznámenie sa s prostredím
- **1. deň:** inštalácia potrebných programov
- **2. – 5. deň:** cvičná úloha
- **6. – 10. deň:** projekt Hedurio
- **11. – 13. deň:** projekt Topstone
- **14. – 32. deň:** projekt požičovňa krojov
- **33. – 50. deň:** projekt MPM management

4.2 Oboznámenie sa s prostredím

Prešiel som školením, zoznámil som sa s pracovníkmi firmy a so systémom na spravovanie úloh, ktorý firma využíva. Všetky úlohy sú zadávané do online systému, ku ktorému majú pracovníci firmy prístup. Nachádza sa tu prehľad úloh a zadávanie jednotlivých úloh. V komentároch jednotlivých úloh sa rozoberá postup práce. Po vykonaní úlohy sa úloha označí za splnenú a zadá sa čas strávený jej vykonávaním. Je to jednoduchý a účinný spôsob spravovania úloh, ktorý mi mnohokrát uľahčil prácu.

4.3 Inštalácia Ruby, Ruby on Rails

Nižšie uvedené programy som inštaloval na platforme Ubuntu 14.04. Ruby som Inštaloval [16] vo verzii 2.1.2p95 vydané 8.5.2014.

Pred inštaláciou samotného Ruby bolo potrebné nainštalovať závislosti a knižnice ako napr. Git, databázu SQLite. Ďalšia závislosť bola **RVM**, pomocou ktorej som následne nainštaloval Ruby príkazom `rvm install 2.1.2p95` .

RVM (Ruby Version Manager) je nástroj príkazového riadku, ktorý umožňuje inštalovať a spravovať Ruby prostredie. Pomocou neho môžme nainštalovať viac verzií Ruby a meniť aktuálne používanú verziu pomocou príkazu `rvm use 2.1.2p95` . V prípade, že chceme danú verziu nastaviť ako predvolenú, tak za verziu pridáme `--default` . Príkazom `rvm list` zobrazíme aktuálne používanú verziu a list všetkých nainštalovaných verzií.

Pred inštaláciou Rails bolo najskôr potrebné nainštalovať NodeJS, ktorý je potrebný pre Coffeescript a ďalšie závislosti (Asset Pipeline). Potom nasledovala inštalácia samotného Rails, ktoré som nainštaloval vo verzii 4.1.4 pomocou príkazu `gem install rails -v 4.1.4` .

Pre prácu v Ruby on Rails som pre jednoduchosť a nízke hardvérové nároky využíval textový editor Geany 1.23.1 s potrebnými rozšíreniami pre Ruby syntax.

Ruby on Rails štandardne využíva databázu SQLite. Tá je ale vhodná len na menšie projekty a v praxi sa častejšie používa MySQL. Aj preto som si vybral MySQL. Pre nainštalovanie MySQL bolo potrebné nainštalovať MySQL server a klienta. Pre prácu s databázou MySQL som si vybral MySQL Workbench, pretože je to pokročilý nástroj a naviac je zadarmo. Nainštaloval som si ho vo verzii 6.0.

4.4 Cvičná úloha

Vytvoril som aplikáciu s názvom blog. Je to jednoduchá webová stránka. Nachádzajú sa tu príspevky, kde každý príspevok môže mať viac komentárov. V Ruby on Rails to zapíšeme ako vzťah `has_many` . V danom prípade sa bude v modeli Post nachádzať `has_many :comments` . V modeli Comment bude `belongs_to :post` . Post má atribúty title (nadpis) a text, Comment má atribúty post_id (id príspevku) a text. Pomocou scaffoldingu sa vygenerujú všetky potrebné modely, view a controllery. Stačilo mi teda upraviť view a jednoduchá aplikácia bola hotová.

```
rails generate scaffold Post title:string text:text
rails generate scaffold Comment post:references text:text
```

Výpis 11: Použité príkazy scaffoldingu

4.5 Projekt Hedurio

Hedurio [17] je unikátny komplexný informačný systém pre riadenie bezpečnostných agentúr. Hedurio efektívne poskytuje prehľad o všetkých činnostiach, umožňuje každému podniku komplexnú správu svojho programu a v mnohých ohľadoch prevyšuje aj požiadavky bezpečnostných agentúr.

Hedurio bol už funkčný informačný systém. Tento projekt mi bol pridelený hlavne preto, aby som dokázal zorientovať sa v cudzom kóde a vytvoriť aj pokročilejší vzhľad.

Mojou úlohou bolo vytvoriť kategóriu „features“ na tejto webovej stránke. Nastavil som, aby bolo možné jednotlivé príspevky spolu s obrázkami pridávať z administrátorského rozhrania. Ďalej som nastavil, aby sa farby pozadia jednotlivých príspevkov striedali. To som dosiahol pomocou Rails metódy cycle, kde sa striedali jednotlivé CSS triedy, ktoré držali jednotlivé farby pozadia.

Keď bola kategória „features“ funkčná, tak som pomocou administrátorského účtu pridal jednotlivé príspevky s obrázkami. Okrem toho som robil malé zmeny vo vzhľade webovej stránky. Na stránku som pridal referencie, ohlasy a partnerov projektu.

4.6 Projekt Topstone

Topstone [18] je tvorca originálnych kamenných povrchov už od roku 1996. Topstone pôsobí a svojou kvalitou je známy v Českej republike, na Slovensku a v Poľsku.

Na tejto webovej aplikácii mi pracovníci Railsformers s.r.o. udelili administrátorské oprávnenia, ktoré mi umožnili administráciu a pridávanie článkov. Pridal som články na zadané témy „svetoznáma lavička Václava Havla v Prahe“, „týždeň otvorených dverí“. Okrem toho som vytvoril slovenský preklad pre túto webovú aplikáciu. Aplikáciu som prekladal pomocou **Rails Internationalization (I18n) API**.

4.7 Projekt požičovňa krojov Ostravica

Požičovňa krojov Ostravica je informačný systém na evidenciu krojov patriacim súboru Ostravica. Na tomto rozsiahlejšom projekte som pracoval samostatne.

Do systému sa evidujú kroje, ktoré sa nachádzajú na sklade, odkiaľ si ich môžu registrovaní užívatelia zapožičať. Evidujú sa informácie o krojoch, kto a kedy si daný kroj zapožičal, kedy bol vrátený a kde sa práve nachádza (či je zapožičaný, na sklade alebo vyradený).

4.7.1 Databáza

Projekt som začal návrhom databáze. Na databáze je založený celý systém, preto ju bolo treba správne navrhnúť.

Schéma vytvorených databázových tabuliek:

Názov atribútu	**Typ**	**Dĺžka**	**Kľúč**	**Povinný**	**Popis**
id	integer		primárny	áno	
area	string	50		áno	názov oblasti

Tabuľka 2: Area_lists – oblasti

Názov atribútu	**Typ**	**Dĺžka**	**Kľúč**	**Povinný**	**Popis**
id	integer		primárny	áno	
folder	string	50		áno	názov adresáru

Tabuľka 3: Folder_lists – adresáre

Názov atribútu	Typ	Dĺžka	Kľúč	Povinný	Popis
id	integer		primárny	áno	
value	float		primárny	áno	hodnota
member_id	integer		cudzí	nie	účastník
note	text	255		nie	poznámka
period_from	date			áno	od
period_to	date			áno	do

Tabuľka 4: Contributions – príspevky

Názov atribútu	Typ	Dĺžka	Kľúč	Povinný	Popis
id	integer		primárny	áno	
subject_name	string	50		nie	názov predmetu
category_man	string	50		nie	mužský
category_woman	string	50		nie	ženský
category_neutral	string	50		nie	neutrálny
area_list_id	integer		cudzí	áno	oblasť
date_of_renting	date			áno	dátum zapožičania
date_of_return	date			nie	dátum vrátenia
on_stock	boolean			áno	na sklade
borrowed	boolean			nie	zapožičané
discarded	boolean			nie	zrušené
member_id	integer		cudzí	áno	účastník
note	text	255		nie	poznámka

Tabuľka 5: Costumes – kroje

Názov atribútu	Typ	Dĺžka	Kľúč	Povinný	Popis
id	integer		primárny	áno	
expertise	string	50		áno	názov zručnosti

Tabuľka 6: Expertise_lists – zručnosti

4.7.2 Postup tvorby aplikácie požičovňa krojov

Databázu som mal navrhnutú, ale aby som databázové tabuľky dostal do aplikácie, bolo treba vytvoriť modely s databázovými migráciami. Tie som vytvoril pomocou príkazu `rails g model NazovModelu nazov_atributu:datovy_typ` s atribútmi podľa navrhnutej databázy. Aby boli vzťahy medzi tabuľkami zachované aj v aplikácii, pridal som tieto vzťahy do modelov.

```
class AreaList < ActiveRecord::Base
    has_many :costumes
    has_many :members
end

class Contribution < ActiveRecord::Base
    belongs_to :member
end

class Costume < ActiveRecord::Base
    belongs_to :area_list
    belongs_to :member
end

class FolderList < ActiveRecord::Base
    has_many :members
end
```

Výpis 12: Určenie vzťahov v modeloch

```
class ExpertiseList < ActiveRecord::
    Base
    has_many :members
end

class Member < ActiveRecord::Base
    has_many :costumes
    has_many :area_lists
    has_many :contributions
    belongs_to : expertise_list
    belongs_to : folder_list
end
```

Výpis 13: Určenie vzťahov v modeloch

Pre oddelenie administrátorského rozhrania od užívateľského som si vytvoril menný priestor s názvom „admin“. Všetky views a controllers menného priestoru sa tak budú nachádzať oddelené od ostatných v `app/views/admin` , resp. v `app/controllers/admin` .

```
namespace :admin do
    resources :users
end
```

Výpis 14: Vytvorenie menného priestoru v súbore config/routes.rb

Ostatné zdroje (resources) som pridal podľa návrhu databázy.

Databáza bude obsahovať veľký počet záznamov, preto bolo požiadavkou vytvorenie **vyhľadávacích formulárov** pre jednoduchšie vyhľadávanie záznamov. Sú to jednoduché formuláre, po ktorých vyplnení sa zobrazia len záznamy splňujúce dané podmienky. Na vytváranie vyhľadávacích formulárov som použil gem 3.1.2 **Ransack**. Ransack má pomocnú metódu **search_form_for**, pomocou ktorej sa dajú jednoducho vytvárať vyhľadávacie formuláre. Dajú sa tak vytvoriť polia na vyhľadávanie podľa názvu, dátumu, čísla, atď.

Pre vyhľadávanie Ransack používa prípony na určenie, ktoré záznamy vyhovujú vyhľadávacím podmienkam. Napr. `= f.search_field :name_cont` vráti všetky záznamy, ktorých názov obsahuje text, ktorý sme zadali do vyhľadávacieho poľa. Naproti

tomu `= f.search_field :name_start` vráti všetky záznamy, ktorých názov začína na zadaný text. Prípony sú v tomto prípade cont (z angl. contains) a starts (z angl. starts with). Ďalšie prípony sú end (ends with), eq (equals), not_ex (not equals), lt (less than), lteq (less than or equal to), in (hodnota sa nachádza v danom zozname), true (záznamy, u ktorých je podmienka pravdivá), false (záznamy, u ktorých je podmienka nepravdivá).

```
>> User.ransack(first_name_cont: 'Rya').result.to_sql
=> SELECT "users".* FROM "users" WHERE ("users"."first_name" LIKE '%Rya%')
```

Výpis 15: Ransack prípona cont v podobe SQL príkazu

Zároveň som pre väčšiu prehľadnosť v záznamoch pridal do tabuliek **usporiadanie záznamov podľa atribútu**. Použil som k tomu Ransack metódu **sort_link**. Pomocou =sort_link(@q, :name) som v hlavičke tabuľky nastavil odkaz. Po kliknutí na daný odkaz sa záznamy tabuľky usporiadajú podľa daného atribútu vzostupne, pri ďalšom kliknutí zostupne. Ak chceme, aby sa záznamy najskôr usporiadali zostupne, tak pridáme do sort_link metódy parameter default_order: :desc. Pre usporiadanie podľa viacerých atribútov naraz použijeme
`=sort_link(@q, :last_name, [:last_name, 'first_name asc'])`.

Aby sa stránky pri veľkom počte záznamov nestali príliš dlhými, bolo potrebné nastaviť **stránkovanie** záznamov. Pod stránkovaním sa myslí zobrazovanie len určitého počtu záznamov na jednej stránke. Ďalšie záznamy zobrazíme pomocou odkazov. Stránka je tak kratšia a rýchlejšie sa načítava. K stránkovaniu som použil gem **will_paginate**.

```
@members = @members.paginate(page: params[:page], per_page: 20)
```

Výpis 16: Nastavenie zobrazovania 20 užívateľov na stránke

Po nastavení stránkovania v controlleri som pridal stránkovanie do view pre jeho zobrazenie na stránke. Pomocou `page_entries_info` sa zobrazí počet zobrazených záznamov z celkového počtu záznamov.

```
.digg_pagination
  = will_paginate @members
  = page_entries_info @members
```

Výpis 17: Pridanie stránkovania vo view

Jedným z požiadaviek bolo, že k aplikácii majú mať prístup len registrovaný užívatelia. Vyžadované boli teda autentifikácia aj autorizácia. Pre **autentifikáciu** (overenie užívateľa) som použil gem **Devise**. Devise sa inicializuje príkazom
`rails generate devise:install`.

Príkazom `rails generate devise User` som vytvoril model User, ktorí som ďalej používal pre autentifikáciu. Tento príkaz zároveň vygeneroval pomocné metódy `user_signed_in?` na zistenie, či je užívateľ prihlásený a `current_user` na zistenie aktuálneho užívateľa. V prípade, že by sme model nazvali Member, tak by sa vygenerovali metódy `current_member` a pod. Po zadaní týchto príkazov boli vytvorené v angličtine aj všetky potrebné view na prihlásenie, registráciu, obnovenie hesla a

emaily na potvrdenie registrácie, obnovenie hesla a iné. Tieto súbory som doplnil o českú lokalizáciu. Ako prihlasovacie údaje som nastavil email a heslo.

Po vyplnení registrácie užívateľom, užívateľ dostane email s pokynmi k potvrdeniu registrácie. Po potvrdení je užívateľ úspešne registrovaný a môže sa do systému prihlásiť. Pre povolenie potvrdzovania registrácie som do modelu User pridal `devise :confirmable` a do databázy som pridal automaticky vygenerované atribúty užívateľa.

Aby sa potvrdzovacie emaily odosielali správne, nastavil som štandardné URL pre vývojové a produkčné prostredie. U vývojového prostredia som nastavil port 3000, pretože rovnaký používa aj Mailcatcher 3.7, ktorý som ďalej používal na testovanie odoslaných mailov. Testoval som či sa emaily správne odoslali, predmet, text, odosielateľa a prijímateľa správy.

```
config.action_mailer.default_url_options = { host: 'localhost', port: 3000 }
```

Výpis 18: Nastavenie štandardného URL pre odosielané emaily v súbore config/environments/development.rb

Email odosielateľa som nastavil takto: `config.mailer_sender = 'noreply@ostravica.com'` .

Ďalšou z podmienok systému bola **autorizácia**. Robil som ju pomocou gemu **CanCanCan**. Príkazom `rails g cancan:ability` sa vytvoril model Ability. V tomto modeli som pomocou `can :manage, :all` určil, že užívateľ môže spravovať všetky objekty (:all), tj. môže vykonávať akékoľvek akcie (:read, :create, :update, :destroy a prípadne ďalšie vlastné akcie). Pomocou `can :read, :all` som ďalej určil, že daný užívateľ môže zobraziť všetky objekty, tj. vykonávať akcie index a show.

```
if user.admin?
  can :manage, :all
elsif user.viewer?
  can :read, : all
```

Výpis 19: Definovanie rolí v models/ability.rb

Pre testovanie oprávnení užívateľa bolo potrebné vytvoriť role. Do tabuľky „users“ som pridal atribút s názvom „role“ typu integer. Pre pridanie, odobranie role užívateľovi a pre testovanie aktuálnej role som vytvoril enumerátor `enum role: {viewer: 0, admin: 1}` .

Ďalšou požiadavkou bolo **generovanie PDF** pre záznamy. Okrem toho, že si užívateľ môže záznamy otvoriť vo webovej stránke HTML, ich môže otvoriť aj vo formáte PDF pre ďalšie použitie (napr. tlač). Pre generovanie PDF som použil gem **Prawn 1.2.1**. PDF som vytváral pre viac modelov, ale v tomto príklade ukážem generovanie pre model Contributions. Ako prvé som nastavil v controlleri, aby metóda index odpovedala na formát PDF: `respond_to :pdf, only: :index` . Všetky metódy teda odpovedajú na formát HTML, index odpovedá aj na formát PDF.

Vo view som pre príspevky nastavil rozloženie strany na šírku, pretože tabuľka môže byť veľká do šírky. Štýl písma som nastavil na Verdana, pretože pri pôvodnom písme finálne PDF nezobrazovalo diakritiku.

Nastavil som, aby na začiatku PDF bol nadpis. Po ňom nasledovala tabuľka. Pred vykreslením tabuľky som si najskôr do premennej uložil prvý riadok (názvy stĺpcov), potom všetky záznamy.

Nasledovalo samotné vykreslenie tabuľky.

```
pdf.table(items, header: true, position: :center) do
 row(0).border_width = 2
end
```

Výpis 20: Vykreslenie tabuľky v views/contributions/index.pdf.prawn

U člena som nastavil **validáciu** pre meno, aby bolo povinné a jeho dĺžka najviac 20 znakov. Ďalej som spravil validáciu PSČ a tel. č. pomocou regulárneho výrazu.

U príspevkov som nastavil validáciu, aby bola hodnota kladná. Maximálnu dĺžku poznámky som nastavil na 255 znakov. Validáciu dátumov som robil pomocou gemu **date_validator**. Nastavil som, aby dátum ukončenia musel byť neskorší ako dátum začiatku.

```
validates :period_to,
   date: { after: :period_from, message: t("validate.date.gt") },
   allow_blank: true
```

Výpis 21: Validácia dátumu v models/contribution.rb

Po tom ako sú modely nastavené sú na rade controllery. Sú potrebné pre prepojenie modelu a view. Model a view prepojíme tým, že si do controlleru uložíme premenné z modelu. Tieto premenné budeme používať vo view a v controlleri samotnom. Ďalej tu definujeme pre každú metódu potrebné správanie a presmerovanie. V controlleri sa použitím CanCanCan metódy **load_and_authorize_resource** načíta pre index `@contributions = Contribution.all` a pre ostatné metódy (show, edit, update, destroy) `@contribution = Contribution.find(params[:id])` , takže už nie je potrebné načítavať aktuálny objekt. Tým aj vo väčšine prípadov odpadá definovanie index, show, edit metód. Stačí teda naimplementovať metódy create, update, destroy. Metóda získava automaticky parametre z metódy resource_params. Nemusíme tak riešiť ani `@contribution = Contribution.new(resource_params)` v metóde create.

```
def update
   @contribution.update(resource_params)
   respond_with @contribution
end

def destroy
   @contribution.destroy
   respond_with :contributions
end
```

Výpis 22: Metódy update, destroy v controllers/contributions_controller.rb

```
def create
   if @contribution.save
      respond_with @contribution
   else
      respond_with [:new, :contribution]
   end
end
```

Výpis 23: Metóda create v súbore controllers/contributions_controller.rb

Tieto metódy vyžadujú metódu `resource_params` pre určenie požadovaných parametrov. Túto metódu som spravil privátnou, tj. prístupnou iba z danej triedy.

```
private
  def resource_params
    params.require(:contribution).permit(:value,  :member_id, :note, :period_from, :period_to)
  end
```

Výpis 24: Metóda resource_params v controllers/contributions_controller.rb

U príspevkov som vo view v indexe vytvoril tlačidlo na otvorenie PDF so všetkými záznamami na novej karte. Ďalej sa tu nachádza tabuľka. V hlavičke tabuľky som pomocou partialu (tj. súboru obsahujúceho časť kódu, ktorý môžeme opakovane používať) vygeneroval sort_link pre každý atribút. V tele tabuľky sú samotné dáta, taktiež v partiale. Pre každý záznam sa tu nachádza tlačidlo zobraziť, zobraziť v PDF, upraviť, odstrániť. Každé tlačidlo má vlastnú ikonu z Font Awesome 3.6. Pod tabuľkou je časť so stránkovaním pomocou will_paginate.

Stránka na zobrazenie daného záznamu (show) zobrazuje všetky tlačidlá pre prácu so záznamom tak, ako v tabuľke pri index stránke. Ďalej sú tu všetky hodnoty daného záznamu.

Stránky na úpravu a vytvorenie nového záznamu (edit a new) vyzerajú rovnako, pretože obidve zobrazujú formulár na zadanie atribútov záznamu. Jediný rozdiel je v nadpise stránky a texte tlačidiel.

S malými zmenami vyzerajú aj view pre ostatné domény (oblasti činnosti, ktoré riešime v rámci informačného systému).

4.8 Projekt MPM

Mojím ďalším samostatným projektom je Market Performance Management (MPM). Je to informačný systém spoločnosti MgC Group. MgC Group je vzdelávacou spoločnosťou, dlhodobo pôsobiaca v ČR, na Slovensku, v Poľsku, Maďarsku, Rakúsku a v Nemecku. Cieľom firmy je pripraviť pre klientov rozvoj zamestnancov, zaistiť implementáciu znalostí do každodennej praxe a nastaviť ich dlhodobú udržateľnosť v praxi.

Systém MPM bude slúžiť k spravovaniu aktivít zamestnancov spoločnosti. Mojou úlohou bolo analyzovať, navrhnúť a naimplementovať tento informačný systém.

4.8.1 Požiadavky

Požiadavky na systém boli nasledovné:

- zabezpečený prístup
- úrovne prístupu (role)
- kalendár aktivít
- súborový systém
- prehľad užívateľov, klientov a ich pobočiek

4.8.2 Postup vorby aplikácie MPM

Databázu som navrhol a následne vytvoril rovnakým postupom ako pri predchádzajúcom projekte. Nachádzajú sa tu databázové tabuľky activities (aktivity), activity_materials (materiály aktivity), branches (pobočky), clients_users (väzobná tabuľka medzi klientmi a užívateľmi), people (kontaktné osoby).

Do modelov som pridal vzťahy z databázy:

```
class Activity < ActiveRecord::Base
  belongs_to :branch
  belongs_to :user
  has_many :activity_materials
end
class ActivityMaterial < ActiveRecord::
    Base
  belongs_to : activity
end
class Branch < ActiveRecord::Base
  belongs_to : client
  has_many :people
  has_many :activities
end
```

Výpis 25: Určenie vzťahov v modeloch

```
class Client < ActiveRecord::Base
  has_many :branches
  has_many :people
  has_many :users, through: :clients_users
end
class Person < ActiveRecord::Base
  belongs_to : client
  belongs_to :branch
end
class User < ActiveRecord::Base
  has_many :activities
  has_many :clients_users
  has_many :clients, through: :
      clients_users
end
```

Výpis 26: Určenie vzťahov v modeloch

Jednou z požiadaviek systému bol **zabezpečený prístup**. Rovnako ako pri predchádzajúcom projekte som použil Devise pre autentizáciu. Postupoval som rovnako, avšak pre autorizáciu som použil okrem CanCanCan aj gem Rolify. Príkazom `rails g rolify Role User` som vytvoril migrácie pre role. Do súboru db/seeds.rb som si potom uložil všetky používané role. Pomocou tohto súboru zaplníme databázu pred tým, než sa aplikácia začne používať.

```
[:admin, :project_manager, :lector, :client, :partner].each do |role|
  Role.create(name: role.to_s)
end
```

Výpis 27: Pridanie rolí do db/seeds.rb

V aplikácii sú tak **úrovne prístupu** (role) admin, projektový manažér, lektor, klient, partner. Každú rolu bližšie popíšem.

Admin môže spravovať všetko.

```
if user.has_role? :admin
  can :manage, :all
```

Výpis 28: Určenie prístupových práv admina v app/models/ability.rb

Projektový manažér môže zobraziť aktivity a jeho klientov, pobočky patriace jeho klientovi. Ďalej môže upravovať svojich klientov a pridávať im pobočky, vytvárať kontaktné osoby, spravovať materiály aktivít, upravovať svoj vlastný účet.

Lektor môže vykonávať akcie definované v metóde **can_client** 4.8.2. Naviac môže upravovať a vytvárať materiály aktivít.

Klient môže vykonávať akcie definované v metóde **can_client** 4.8.2 a naviac zobrazovať materiály aktivity pod jeho správou.

Partner môže zobrazovať vlastné pobočky, klientov a aktivity jeho vlastných pobočiek, kontaktné osoby, materiály aktivít pod jeho správou a upravovať svoj vlastný účet.

U rolí v súbore `app/models/ability.rb` som vytvoril metódu **can_client** , aby sa neopakoval zdrojový kód, keďže lektor a klient majú podobné práva. Zdieľajú preto túto metódu. Môžu zobrazovať vlastných klientov a pobočky, kontaktné osoby, vlastné aktivity a list týchto aktivít. Ďalej môžu upravovať svoj vlastný účet.

Pre **výber dátumu a času** som použil **jquery-datetimepicker-rails**. Pre jeho fungovanie bolo treba pridať potrebné CSS a JavaScript moduly do
`app/assets/stylesheets/application.css` pre CSS, resp.
`app/assets/javascripts/application.js` pre JavaScript.

Do súboru application.js som pridal funkciu na pridanie vkladania dátumu a času (datetimepicker), nastavil som formát výsledného reťazca, jazyk na češtinu a 15 minútový rozdiel medzi jednotlivými časmi.

```
$(function () {
    $('.dtp').datetimepicker({format:'d.m.Y H:i', lang:'cs', step:15});
});
```

Výpis 29: Funkcia na vkladanie dátumu a času v app/assets/javascripts/application.js

Ďalším požiadavkom bol **kalendár s udalosťami**. Udalosťami kalendára sú jednotlivé aktivity užívateľov. Užívateľ si tak môže zobraziť prehľadný rozpis jeho plánovaných aktivít vo forme kalendára s pohľadom na mesiac. Medzi jednotlivými mesiacmi sa dá posúvať. Do kalendára je možné pridávať ďalšie aktivity, prípadne ich upravovať alebo odstraňovať. Pre túto úlohu som použil **fullcalendar**.

V modeli aktivity som si vytvoril obmedzenie s názvom „between“, ktoré vyberie všetky záznamy, v ktorých je dátum začiatku (start_time) menší ako dátum ukončenia (end_time). Ďalej som si vytvoril pole, ktoré drží RGB hodnoty farieb pre jednotlivé stavy aktivity (plánovaná, potvrdená, zaplatená). Aktivity v kalendári tak majú odlišné farby podľa ich aktuálneho stavu.

```
scope :between, ->(start_time, end_time) { where("? < start_time < ?", start_time.to_date,
    end_time.to_date) }

EVENT_COLORS = { planned: "#ce130e", accepted: "#5b6df6", invoiced: "#4c941f" }
EVENT_COLORS.default = :planned
```

Výpis 30: app/models/activity.rb

V controlleri v index metóde som si do premennej uložil záznamy z vopred vytvoreného obmedzenia „between“. Záznamy z tohto obmedzenia budú zobrazované v kalendári.

Pre uloženie záznamov, ktoré budú vykresľované do kalendára bolo ďalej potrebné vytvoriť JSON pole. To som vytvoril vo view pomocou jbuilder. Pole obsahuje jednotlivé

atribúty záznamov a farby udalostí v kalendári. Fullcalendar požaduje atribúty ako title, start, end, a preto bolo potrebné premenovať vyhovujúce atribúty modelu.

```
json.array!(@activities) do | activity |
  json.extract! activity , :id, :branch_id, :user_id
  json. title  activity .full_name
  json. start  activity .start_time
  json.end activity .end_time
  json. url  activity_url ( activity )
  json.color  Activity :: EVENT_COLORS[activity.status.to_sym]
end
```

Výpis 31: Vytvorenie JSON poľa v app/views/activities/index.json.jbuilder

Nasledovalo samotné vykreslenie kalendára s `id="calendar"` vo view. Parametre kalendára som zadal pomocou CoffeeScriptu.

```
#calendar

:coffee
  $ ->
    $('#calendar').fullCalendar
      buttonText:
        today: $.i18n._('today')
      events: window.location.href
      firstDay : 1
      header: { left : '', center: ' title ', right : 'today prev,next' }
```

Výpis 32: app/views/activities/index.html.haml

Ďalším požiadavkom bolo vytvorenie **súborového systému**. Využil som k tomu gem **Carrierwave**, keďže je pre tento účel najpoužívanejší. Ako prvé bolo potrebné vygenerovanie nahrávača súborov (uploader) pomocou príkazu `rails generate uploader Material`. Ten sa následne uložil do `app/uploaders/material_uploader.rb`. Nahrávač súborov zabezpečuje samotné načítanie a uloženie súboru. Pomocou spojovacej tabuľky activity_materials som umožnil, aby jedna aktivita mohla mať viac materiálov. V modeli ActivityMaterial som nastavil nahrávač súborov pomocou `mount_uploader :material, MaterialUploader`.

Carrierwave rieši získanie súboru za nás, a tak pre nahrávanie súboru stačí vo view použiť metódu `input` pre simple_form_for, resp. `file_field` pre form_for. V tomto prípade som použil `= f.input :material`.

Nasleduje samotné zobrazenie materiálu vo view a odkaz na jeho stiahnutie. Pomocou `activity_material.material` zobrazíme názov súboru. Odkaz na stiahnutie súboru sa zobrazuje tak, že za objekt pridáme „.url“, v tomto prípade `activity_material.material.url`.

4.8.3 Testovanie

K testovaniu som použil gemy **rspec-rails** a **factory_girl_rails**, ktoré som povolil len pre vývojové a testovacie prostredie (v produkčnom prostredí nie sú potrebné). Príkazom `rails generate rspec:install` sa vytvorí adresár, v ktorom sa budú nachádzať jednotlivé testy. Pomocou príkazu `rspec spec` sa potom spustia všetky testy, `rspec spec/models` spustí vykonávanie testov len pre modely a pod.

Databázová schéma nie je tá istá, ktorá sa používa na vývoj. Býva prázdna, pri testoch sa vytvoria potrebné záznamy a po testoch sa automaticky vyprázdni. Aby sme mohli aplikáciu testovať, potrebujeme nejaké dáta, na ktorých budeme testy vykonávať. Tieto dáta si vytvoríme pomocou gemu **FactoryGirl**.

```
FactoryGirl.define do
  factory :activity do
    title "aktivita1"
    short_name "a1"
    start_time "2015-02-12 19:45:00"
    end_time "2015-02-12 20:00:00"
    branch { Branch.first || create(:branch) }
  end
end
```

Výpis 33: Vytvorenie testovacieho objektu v spec/factories/activities.rb

Po vytvorení testovacieho objektu môžme prejsť na samotné testovanie. Začneme testovaním modelu. U aktivity som testoval vzťahy modelu. Na test či aktivita patrí užívateľovi som použil nasledovný riadok kódu: `it {should belong_to(:branch)}` . Ak očakávame, že pri vymazaní objektu sa vymaže aj objekt vo vzťahu, tak za zátvorku s názvom tabuľky pridáme `.dependent(:destroy)` .

Pri testovaní modelu testujeme aj validácie. Ak má byť atribút povinný, tak testujeme pomocou `validate_presence_of` . Ja som v modeli aktivity testoval povinnosť týchto atribútov: title, short_name, branch, start_time, end_time.

Nasleduje testovanie controlleru. Tu som si najskôr vytvoril metódu, ktorá sa spustí pred každým testom daného controlleru. Táto metóda vytvorí užívateľa, prihlási ho do systému a vytvorí objekt, ktorý sme si vopred definovali vo FactoryGirl.

```
before :each do
  @user = FactoryGirl.create(:user, :admin)
  sign_in(@user)

  @activity = FactoryGirl.create(: activity )
end
```

Výpis 34: spec/controllers/activities_controller_spec.rb — metóda, ktorá sa spustí pred každým testom

Nasleduje testovanie samotných akcií controlleru. V indexe testujeme, či sa v ňom nachádza objekt zo spomínanej metódy, ktorá sa spustí pred každým testom. Ďalej som testoval, či sa HTTP odpoveď rovnala „200 OK“, čo znamená, že všetko prebehlo v poriadku a stránka sa zobrazila.

Testy pre metódy show a edit budú vyzerať podobne ako test index, a tak ich preskočíme.

V create metóde testujeme, či sa počet záznamov zvýši o 1 (tzn. záznam sa vytvorí). Ďalej som otestoval správnosť presmerovania po vytvorení.

V update metóde testujeme, či sa hodnoty atribútov zmenili. Zároveň som otestoval, či nás správne presmeruje po uložení zmien.

V destroy metóde testujeme, či sa počet záznamov zmenšil o 1 (tzn. či sa záznam odstránil). Ďalej som otestoval správnosť presmerovania po odstránení. Test bude teda vypadať podobne ako v create metóde.

5 Záver

Na praxi som prvé dva týždne pracoval vo dvojici, zvyšok času som pracoval sám. Naučil som sa tak pracovať samostatne a vyskúšal som si aj prácu v tíme.

Taktiež som si osvojil niektoré už nadobudnuté školské znalosti. Zo školských znalostí som konkrétne uplatnil návrh a prácu s databázami (predmety Úvod do databázových systémov, Databázové a informačné systémy), návrh vzhľadu aplikácie (predmety Užívateľské rozhrania, Vývoj informačných systémov), návrhové vzory (predmet Vývoj informačných systémov), HTML, CSS, JavaScript, HTTP metódy GET, POST, atď. (predmet Vývoj internetových aplikácií). V menšej miere som uplatnil aj znalosti z predmetu Skriptovacie programovacie jazyky a ich aplikácie, kde sme sa učili programovací jazyk Python často porovnávaný s jazykom Ruby. Obidve sú to skriptovacie programovacie jazyky a sú vysokoúrovňové. Zároveň sme sa na tomto predmete učili základy vývoja vo webovom frameworku Django. Vďaka tomu som si prvý krát vyskúšal vývoj webových aplikácií vo webovom frameworku, čo som využil v Ruby on Rails.

Naučil som sa skriptovací programovací jazyk Ruby, tvorbu web. aplikácií v Ruby on Rails, implementáciu návrhových vzorov, testovanie aplikácií, pracovať s najpoužívanejšími rozšíreniami Ruby on Rails, tvorbu súborového systému na nahrávanie a sťahovanie súborov, pracovať s technológiami ako Bootstrap a Git. V neposlednom rade som vďaka praxi získal nové kontakty, s ktorými môžem v budúcnosti v danom odbore spolupracovať.

Získal som skúsenosti a množstvo nových vedomostí, ktoré v budúcnosti určite využijem. Všetky zmienené nadobudnuté skúsenosti a vedomosti mi budú prospešné pri výbere práce a následnom uplatnení sa.

6 Referencie

[1] *Railsformers s.r.o.* [online]. [cit. 24. 2. 2015].
Dostupné na `http://railsformers.com/cs`

[2] *Ruby.* [online]. [cit. 24. 2. 2015].
Dostupné na `https://www.ruby-lang.org/en/`

[3] *Ruby on Rails.* [online]. [cit. 24. 2. 2015].
Dostupné na `http://cs.wikipedia.org/wiki/Ruby_on_Rails`

[4] Fowler, Martin, *Patterns of Enterprise Application Architecture* Boston: Addison Wesley, 2002, 560 s. ISBN 0-321-12742-0.

[5] *Obrázok MVC.* [online]. [cit. 24. 2. 2015].
Dostupné na
`http://es.wikibooks.org/wiki/XForms/Arquitectura_MVC`

[6] *HAML.* [online]. [cit. 24. 2. 2015].
Dostupné na `http://haml.info/`

[7] *Obrázok CSS syntaxe.* [online]. [cit. 24. 2. 2015].
Dostupné na `http://www.etsav.upc.edu/assignatures/portafoli/tutorial2/6.html`

[8] *VŠB – stránky predmetu VIA.* [online]. [cit. 24. 2. 2015].
Dostupné na `http://www.cs.vsb.cz/osobni-stranky/michal-radecky/vyvoj-internetovych-aplikaci-via-1.aspx`

[9] *CoffeeScript.* [online]. [cit. 24. 2. 2015].
Dostupné na `http://en.wikipedia.org/wiki/CoffeeScript`

[10] *Front-end.* [online]. [cit. 24. 2. 2015].
Dostupné na `http://www.adaptic.cz/znalosti/slovnicek/frontend/`

[11] *Obrázok Front-end, Back-end.* [online]. [cit. 24. 2. 2015].
Dostupné na
`http://www.tonymarston.net/php-mysql/front-end-back-end-01.png`

[12] *Bootstrap.* [online]. [cit. 24. 2. 2015].
Dostupné na `http://www.w3schools.com/bootstrap/`

[13] *Font Awesome.* [online]. [cit. 24. 2. 2015].
Dostupné na `http://fortawesome.github.io/Font-Awesome/`

[14] *Mailcatcher.* [online]. [cit. 24. 2. 2015].
Dostupné na `http://mailcatcher.me/`

[15] *Git.* [online]. [cit. 24. 2. 2015].
Dostupné na `http://git-scm.com/`

[16] *Setup Ruby On Rails on Ubuntu 14.04 Trusty Tahr.* [online]. [cit. 24. 2. 2015].
Dostupné na `https://gorails.com/setup/ubuntu/14.04`

[17] *Hedurio.* [online]. [cit. 24. 2. 2015].
Dostupné na `http://hedurio.com/`

[18] *Topstone.* [online]. [cit. 24. 2. 2015].
Dostupné na `http://www.topstone-sk.sk/`

7 Zoznam príloh

Obsah priloženého CD:

- Elektronická verzia bakalárskej práce
- snimky_aplikacii – adresár so snímkami výsledných aplikácií
- vystupne_subory – adresár s ostatnými výstupnými súbormi
- zadania – adresár so zadaniami a nákresmi k riešeným úlohám